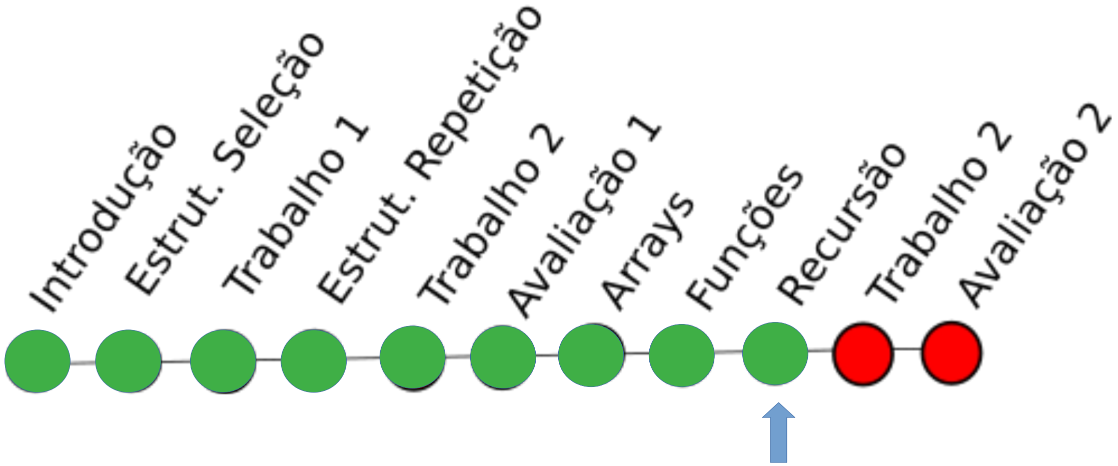
Tópico 08 – Funções

Prof. André Gustavo Hochuli

Plano de Aula

- Definição de Função
- Implementações de Funções
- Exercícios



Recursão

- Uma função recursiva é aquela que faz referência a si própria na sua definição

- Um exemplo clássico é o cálculo fatorial:

- (!4) $=$ (4 x 3 x 2 x 1) $\rightarrow$ 24
  - F(4) $\rightarrow$ 4 x F(3)

$$\begin{cases} 1 & \text{se } n = 0 \\ n \times (n-1)! & \text{se } n > 0 \end{cases}$$

Recursão

- Uma função recursiva é aquela que faz referência a si própria na sua definição

- Um exemplo clássico é o cálculo fatorial:

- (!4) = (4 x 3 x 2 x 1) $\rightarrow$ 24
    - F(4) $\rightarrow$ 4 x F(3)
    - F(3) $\rightarrow$ 3 x F(2)

$$\begin{cases} 1 & \text{se } n = 0 \\ n \times (n-1)! & \text{se } n > 0 \end{cases}$$

Recursão

- Uma função recursiva é aquela que faz referência a si própria na sua definição

- Um exemplo clássico é o cálculo fatorial:

- (!4) = (4 x 3 x 2 x 1) $\rightarrow$ 24
  - F(4) $\rightarrow$ 4 x F(3)
  - F(3) $\rightarrow$ 3 x F(2)
  - F(2) $\rightarrow$ 2 x F(1)

$$\begin{cases} 1 & \text{se } n = 0 \\ n \times (n-1)! & \text{se } n > 0 \end{cases}$$

Recursão

- Uma função recursiva é aquela que faz referência a si própria na sua definição

- Um exemplo clássico é o cálculo fatorial:

- (!4) = (4 x 3 x 2 x 1) $\rightarrow$ 24
  - F(4) $\rightarrow$ 4 x F(3)
  - F(3) $\rightarrow$ 3 x F(2)
  - F(2) $\rightarrow$ 2 x F(1)
  - F(1) $\rightarrow$ 1 x F(0)

$$\begin{cases} 1 & \text{se } n = 0 \\ n \times (n-1)! & \text{se } n > 0 \end{cases}$$

Recursão

- Uma função recursiva é aquela que faz referência a si própria na sua definição

- Um exemplo clássico é o cálculo fatorial:

- (!4) = (4 x 3 x 2 x 1) $\rightarrow$ 24
  - F(4) $\rightarrow$ 4 x F(3)
  - F(3) $\rightarrow$ 3 x F(2)
  - F(2) $\rightarrow$ 2 x F(1)
  - F(1) $\rightarrow$ 1 x F(0)
  - F(0) $\rightarrow$ 1

$$\begin{cases} 1 & \text{se } n = 0 \\ n \times (n-1)! & \text{se } n > 0 \end{cases}$$

Recursão

- Uma função recursiva é aquela que faz referência a si própria na sua definição

- Um exemplo clássico é o cálculo fatorial:

- $(!4) = (4 \times 3 \times 2 \times 1) \rightarrow 24$
  - $F(4) \rightarrow 4 \times F(3)$
  - $F(3) \rightarrow 3 \times F(2)$
  - $F(2) \rightarrow 2 \times F(1)$
  - $F(1) \rightarrow 1 \times 1 \rightarrow 1$
  - $F(0) \rightarrow 1$

$$\begin{cases} 1 & \text{se } n = 0 \\ n \times (n-1)! & \text{se } n > 0 \end{cases}$$

Recursão

- Uma função recursiva é aquela que faz referência a si própria na sua definição

- Um exemplo clássico é o cálculo fatorial:

- (!4) = (4 x 3 x 2 x 1) $\rightarrow$ 24
  - F(4) $\rightarrow$ 4 x F(3)
  - F(3) $\rightarrow$ 3 x F(2)
  - F(2) $\rightarrow$ 2 x 1 $\rightarrow$ 2
  - F(1) $\rightarrow$ 1 x 1 $\rightarrow$ 1
  - F(0) $\rightarrow$ 1

$$\begin{cases} 1 & \text{se } n = 0 \\ n \times (n-1)! & \text{se } n > 0 \end{cases}$$

Recursão

- Uma função recursiva é aquela que faz referência a si própria na sua definição

- Um exemplo clássico é o cálculo fatorial:

- (!4) = (4 x 3 x 2 x 1) $\rightarrow$ 24
  - F(4) $\rightarrow$ 4 x F(3)
  - F(3) $\rightarrow$ 3 x 2 $\rightarrow$ 6
  - F(2) $\rightarrow$ 2 x 1 $\rightarrow$ 2
  - F(1) $\rightarrow$ 1 x 1 $\rightarrow$ 1
  - F(0) $\rightarrow$ 1

$$\begin{cases} 1 & \text{se } n = 0 \\ n \times (n-1)! & \text{se } n > 0 \end{cases}$$

Recursão

- Uma função recursiva é aquela que faz referência a si própria na sua definição

- Um exemplo clássico é o cálculo fatorial:

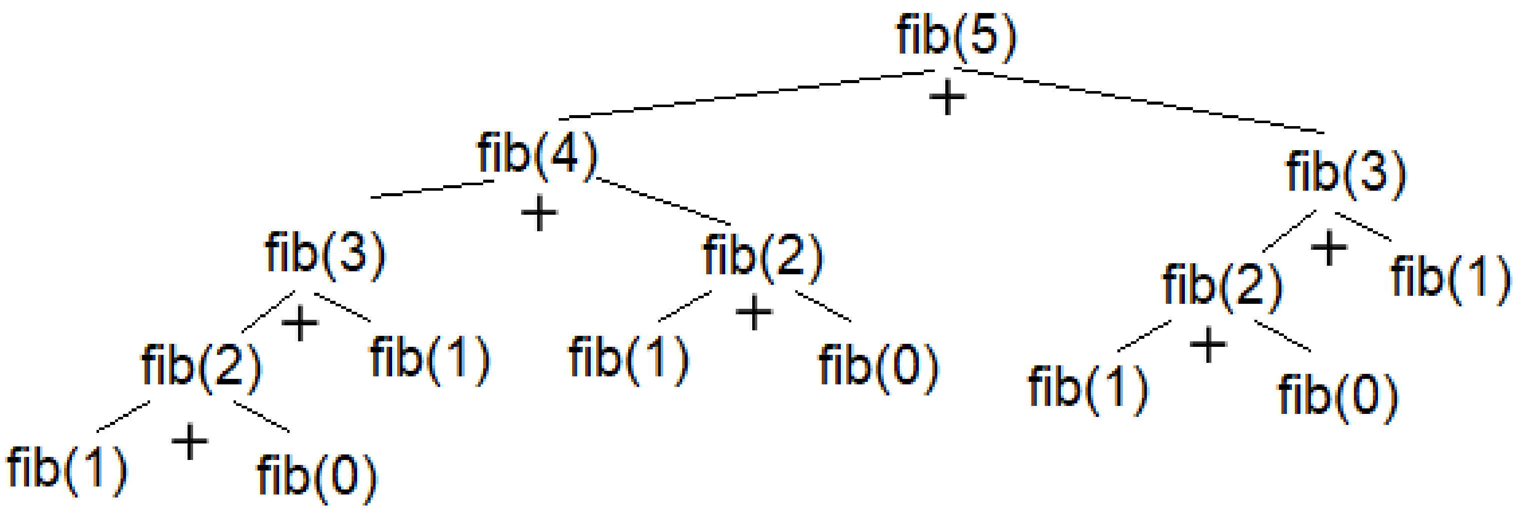
- (!4) = (4 x 3 x 2 x 1) $\rightarrow$ 24
  - F(4) $\rightarrow$ 4 x 6 $\rightarrow$ 24
  - F(3) $\rightarrow$ 3 x 2 $\rightarrow$ 6
  - F(2) $\rightarrow$ 2 x 1 $\rightarrow$ 2
  - F(1) $\rightarrow$ 1 x 1 $\rightarrow$ 1
  - F(0) $\rightarrow$ 1

$$\begin{cases} 1 & \text{se } n = 0 \\ n \times (n-1)! & \text{se } n > 0 \end{cases}$$

Recursão

- Outro exemplo é a série de Fibonacci:

$$\begin{cases} 0 & \text{se } n = 0 \\ 1 & \text{se } n = 1 \\ F(n-1) + F(n-2) & \text{se } n > 1 \end{cases}$$



Recursão

- E na programação?
- (!4) = (4 x 3 x 2 x 1) → 24
  - F(4) → 4 x F(3)
  - F(3) → 3 x F(2)
  - F(2) → 2 x 1 → 2
  - F(1) → 1 x 1 → 1
  - F(0) → 1

$$\begin{cases} 1 & \text{se } n = 0 \\ n \times (n-1)! & \text{se } n > 0 \end{cases}$$

```
def fatorial(n):
    if n == 0:
        return 1
    else:
        return n * fatorial(n - 1)
```

Recursão

- E na programação?
- $(!4) = (4 \times 3 \times 2 \times 1) \rightarrow 24$
  - $F(4) \rightarrow 4 \times F(3)$
  - $F(3) \rightarrow 3 \times F(2)$
  - $F(2) \rightarrow 2 \times 1 \rightarrow 2$
  - $F(1) \rightarrow 1 \times 1 \rightarrow 1$
  - $F(0) \rightarrow 1$

$$\begin{cases} 1 & \text{se } n = 0 \\ n \times (n-1)! & \text{se } n > 0 \end{cases}$$

```python
def fatorial(n):
    if n == 0:
        return 1
    else:
        return n * fatorial(n - 1)
```

- Ao implementar recursão deve-se atentar:
  - Problema Base (Critério de Parada)
  - Convergência para o problema base (sub-problema, ex: (n-1))

Recursão

- E na programação?
- (!4) = (4 x 3 x 2 x 1) $\rightarrow$ 24
  - F(4) $\rightarrow$ 4 x F(3)
  - F(3) $\rightarrow$ 3 x F(2)
  - F(2) $\rightarrow$ 2 x 1 $\rightarrow$ 2
  - F(1) $\rightarrow$ 1 x 1 $\rightarrow$ 1
  - F(0) $\rightarrow$ 1

$$\begin{cases} 1 & \text{se } n = 0 \\ n \times (n-1)! & \text{se } n > 0 \end{cases}$$

```
def fatorial(n):
    if n == 0:
        return 1
    else:
        return n * fatorial(n - 1)
```

- Ao implementar recursão deve-se definir:
  - Problema Base (Critério de Parada)
  - Convergência para o problema base (sub-problema, ex: (n-1))

Recursão

- Vamos codificar alguns exercícios:
  - Print N até 0 (Avaliar print antes e depois do return)
  - Soma de N até 0
  - Soma dos dígitos de um Número
  - Soma de um vetor
  - Fatorial/Fibonacci
  - Selection Sort

    - Siga o roteiro
      - Encontrar o problema base
      - Definir a convergência

Recursão

- Outro exemplo é a série de Fibonacci: